ECE113 Homework 2:

This section is used for the problems of the homework that required code based solutions. This section will be broken down by each problem.

5C:

Python Code:

In this problem we are carrying out the convolution of our equivalent system h_{eq}. This was done in python. As you can see in the code, the individual impulse responses of each block were first created, and then they were combined in the method determined in 5B to create our equivalent system. This system is then convolved with a step input (x[n]). The code written is as follows:

```
import numpy as np
import matplotlib.pyplot as plt

# 5b:
index =np.arange(-20,20)
h2 = np.zeros(len(index))
for i in index:
    if(index[i] <0 or index[i] >= 3):
        h2[i] = 0
    else:
        h2[i] = 1

h3 = h2
h4 =np.zeros(len(index))
for i in index:
    if(index[i] == 2):
        h4[i]=1
    else:
        h4[i] = 0
h_eq_1 = np.convolve(h2,h3)
h_eq_2 = np.convolve (h2, h4)
plot_axis = np.arange(-40,39)
h1 = np.zeros(len(plot_axis))
for i in plot_axis:
    if(plot_axis[i] <0):
        h1[i]  = 0
    else:
        h1[i] = np.exp(-0.1*plot_axis[i])

h_eq = h1 + h_eq_1 +h_eq_2

plt.figure(1)
plt.stem(plot_axis, h_eq, use_line_collection =True)
plt.xlabel("index")
plt.ylabel("IRF")
plt.savefig("hw2_5b.png")

#5c
```

```
x = np.zeros(len(plot_axis))
for i in plot_axis:
    if(i>=0):
        x[40+i] = 1

y = np.convolve(x,h_eq,'same')
truncated_axis = np.arange(0,11)
y_truncated  = np.zeros(len(truncated_axis))
for i in plot_axis:
    if(i>= 0 and i<=10):
        y_truncated[i] = y[40+i]
plt.figure(2)
plt.stem(truncated_axis,y_truncated, use_line_collection=True)
plt.xlabel("index")
plt.ylabel("Output")
plt.savefig("hw2_5c.png")
```

Results:

Figure 1 shows the result of the convolution of x[n] with heq[n]. This is our output y[n] and is plotted below:

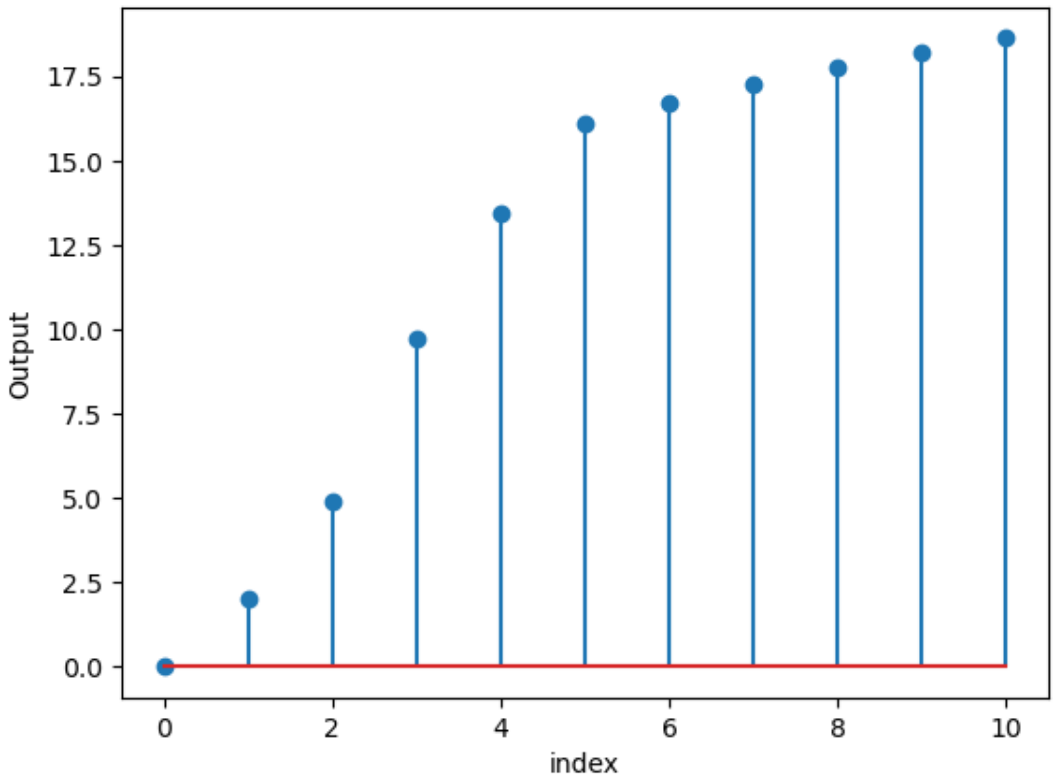

Figure 1: y[n]

6:

This section was more easily implemented in MATLAB so the resulting code is shown below. Note that the code is annotated to show which section of the code pertains to part a and b.

Code:

The code for this problem is as follows:

```
close all;
clear all;
clc;

% Load the sound track
[X, Fs] = audioread('inception_sound_track.wav');

% ------------------------------------------------------------------------------
% Please enter your code here
% Design the system as shown in the figure that performs upsampling followed by a smoother

%6 in general:

Y = zeros( 3*length(X),2);
% Upsampling implementation
for j =1:length(Y)
    if(~mod(j,3))
        Y(j,:) = X(j/3,:);
    end
end

% Smoother using moving average and exponential smoother

% 6a:

%moving average
n=10;
Y1 = zeros( 3*length(X),2);
for i = 1:length(Y)
    sum=[0,0];
    for j =0:n
        if(i-j >=1)
            sum = sum + Y(i-j);
        end
    end
    avg =sum/(n+1);
    Y1(i,:) = avg;
end
%10 best without losing volume

%6b:
%exponential smoother
alpha =0.5;
Y2 = zeros( 3*length(X),2);
```

```
for i = 1:length(Y)
    if(i-1)>=1
        Y2(i,:) = (1-alpha)*Y2(i-1,:) + alpha*Y(i,:);
    end
end
%0.5 appears to be the best


%note that for this code to be run, the output Y1 will be the moving average and Y2...
% will be the exponential smoother

% ------------------------------------------------------------------------------
% Play the output signal Y (y[n]) from the system should be slower

ap_x = audioplayer(Y2, Fs); % Play the output audio file with original sampling frequency
play(ap_x)

% Please attach/print your code to the homework submission
```

Results:

a: The window length that appeared to sound the best for the moving average was a step size of 10. Below 10 the sound still sounds scratchy with peaks that should not be in the sound. This suggests that windows below 10 are not aqequately smoothing the signal. We can also notice that while step sizes of more than ten (50 and 100) are also smooth, they lack volume. All step sizes after 10 began to lose volume without becoming any more smooth. This was likely a result of the peaks of amplitude being averaged out with the quieter parts of the sound. As a result the overall sound felt more muffeled. Distinct beats were less noticeable. As a result 10 was the best window size for our moving average. The code for this section can be seen above labeled 6a.

b: The α that produced the best result was $\alpha = 0.5$. This gave us a smooth sounding audio that maintained volume. An alpha of 0 was not a good solution as our initialized array is at zero state. This resulted in an all zero output–it was not at all based off of the input. An alpha of 0.3 was also not good as it was a very quiet audio, as the input was not weighted heavily. at 0.5 our output was both smooth and loud–it was a good balance of the two. However, at 0.8 the audio began to become more scratchy–it was becoming more like the input. 1 clearly was not the best solution as this produced an output identical to the input–all scratchyness remained. As a result, 0.5 was the best alpha for the system as it produced the sound with the best quality. This makes sense as it equally weights the input and previous values, which will smooth the data points without excessively damping them.